US008156415B1

(12) **United States Patent**
Nguyen et al.

(10) **Patent No.: US 8,156,415 B1**
(45) **Date of Patent: Apr. 10, 2012**

(54) **METHOD AND SYSTEM FOR COMMAND QUEUING IN DISK DRIVES**

(75) Inventors: **Huy Tu Nguyen**, Laguna Hills, CA (US); **William C. Wong**, Cerritos, CA (US); **Kha Nguyen**, Anaheim, CA (US); **Yehua Yang**, Las Flores, CA (US)

(73) Assignee: **Marvell International Ltd.**, Hamilton (BM)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 781 days.

(21) Appl. No.: **12/323,267**

(22) Filed: **Nov. 25, 2008**

Related U.S. Application Data

(60) Provisional application No. 61/016,667, filed on Dec. 26, 2007.

(51) **Int. Cl.**
G06F 11/08 (2006.01)
G06F 13/00 (2006.01)

(52) **U.S. Cl.** **714/807**; 710/5

(58) **Field of Classification Search** 714/807; 710/5

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,789,994	A	12/1988	Randall et al.	
5,909,384	A	6/1999	Tal et al.	
6,170,042	B1 *	1/2001	Gaertner et al.	711/158
6,292,856	B1 *	9/2001	Marcotte	710/39
6,745,266	B1 *	6/2004	Johnson et al.	710/65
6,745,303	B2 *	6/2004	Watanabe	711/161
6,892,250	B2 *	5/2005	Hoskins	710/6
6,925,539	B2 *	8/2005	Mowery et al.	711/158
7,043,567	B2 *	5/2006	Trantham	710/6
7,158,324	B2	1/2007	Stein et al.	
7,194,607	B2 *	3/2007	Dahlen et al.	712/227
7,203,232	B2	4/2007	Ahn	
7,330,068	B2	2/2008	Barksdale	
7,421,614	B2 *	9/2008	Watanabe	714/6.1
7,849,259	B1 *	12/2010	Wong et al.	711/112
7,987,396	B1	7/2011	Riani	
2006/0146926	A1	7/2006	Bhoja et al.	
2007/0101075	A1 *	5/2007	Jeddeloh	711/158
2009/0135035	A1	5/2009	Fifield	

OTHER PUBLICATIONS

"Notice of Allowance", U.S. Appl. No. 12/556,483, (Mar. 29, 2011), 6 pages.
"Notice of Allowance", U.S. Appl. No. 12/120,483, (Nov. 25, 2011), 8 pages.

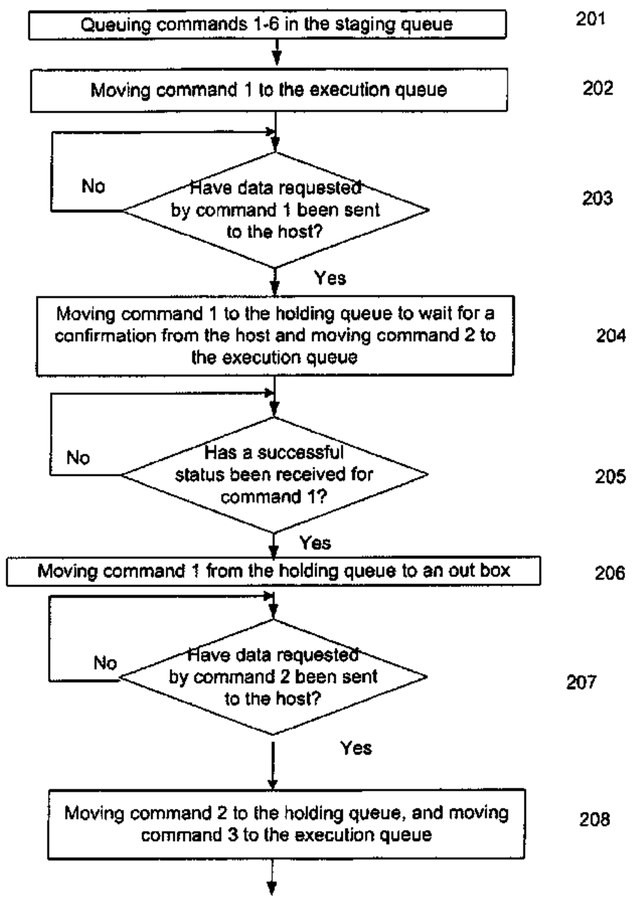
* cited by examiner

Primary Examiner — Stephen Baker

(57) **ABSTRACT**

A method and system for command queuing in disk drives may improve performance by queuing multiple commands and sequentially executing them automatically without firmware intervention. The method may use a number of queues, e.g., a staging queue for commands to be executed, an execution queue for commands currently being executed, and a holding queue for commands which have been executed but have not received a status report from a host. With the pipelined nature of queued commands, when data requested by one command are being sent to the host, the queue logic may already be fetching data for the next command. If an error occurs in the transmission, commands in the queues may backtrack and restart from the point where data were last known to have been successfully sent to the host.

24 Claims, 6 Drawing Sheets



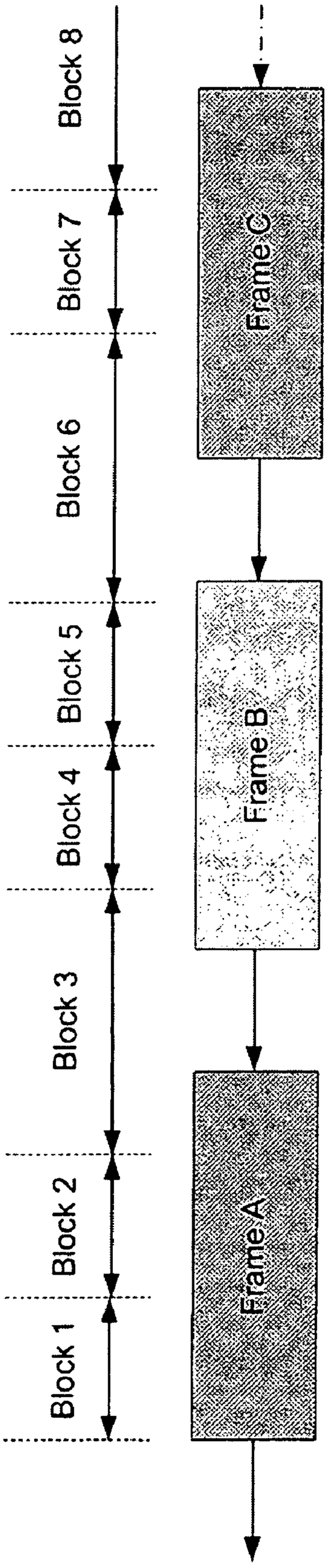

Block 8
Block 7
Block 6
Block 5
Block 4
Block 3
Block 2
Block 1
Frame C
Frame B
Frame A

FIG. 1

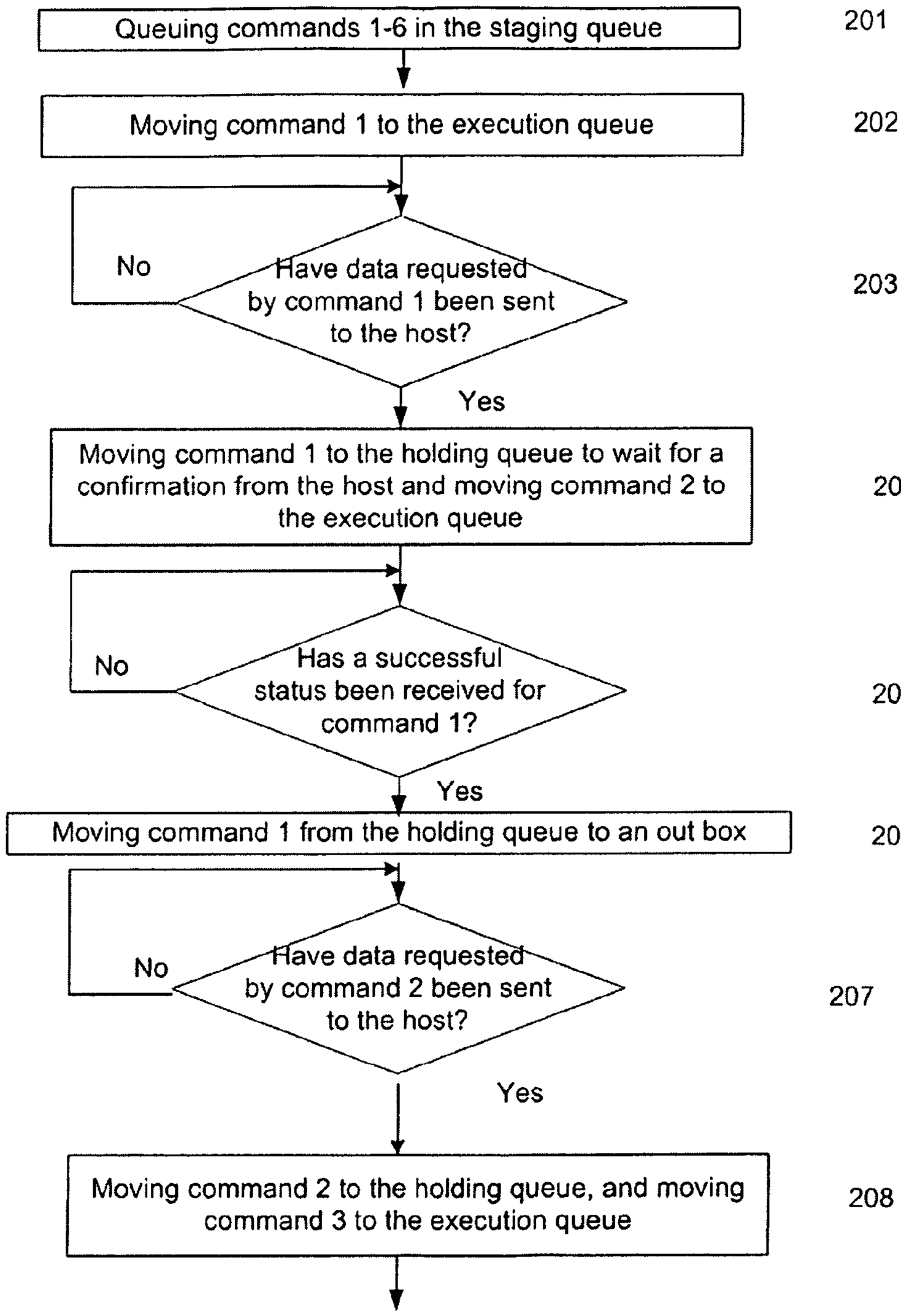

Queuing commands 1-6 in the staging queue
201
Moving command 1 to the execution queue
202
No
Have data requested by command 1 been sent to the host?
203
Yes
Moving command 1 to the holding queue to wait for a confirmation from the host and moving command 2 to the execution queue
204
No
Has a successful status been received for command 1?
205
Yes
Moving command 1 from the holding queue to an out box
206
No
Have data requested by command 2 been sent to the host?
207
Yes
Moving command 2 to the holding queue, and moving command 3 to the execution queue
208

FIG. 2

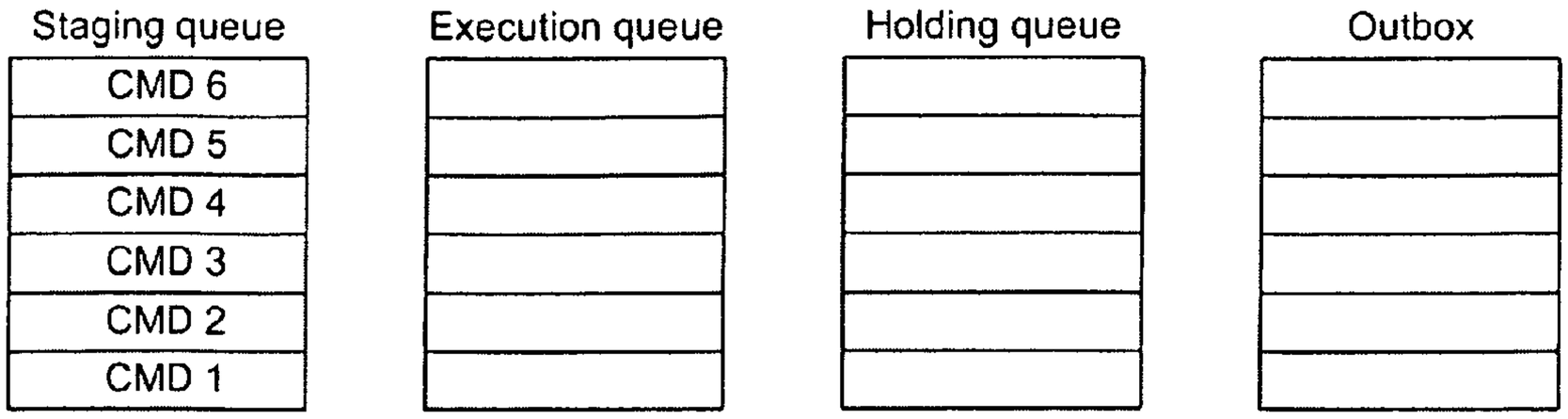


FIG. 3A

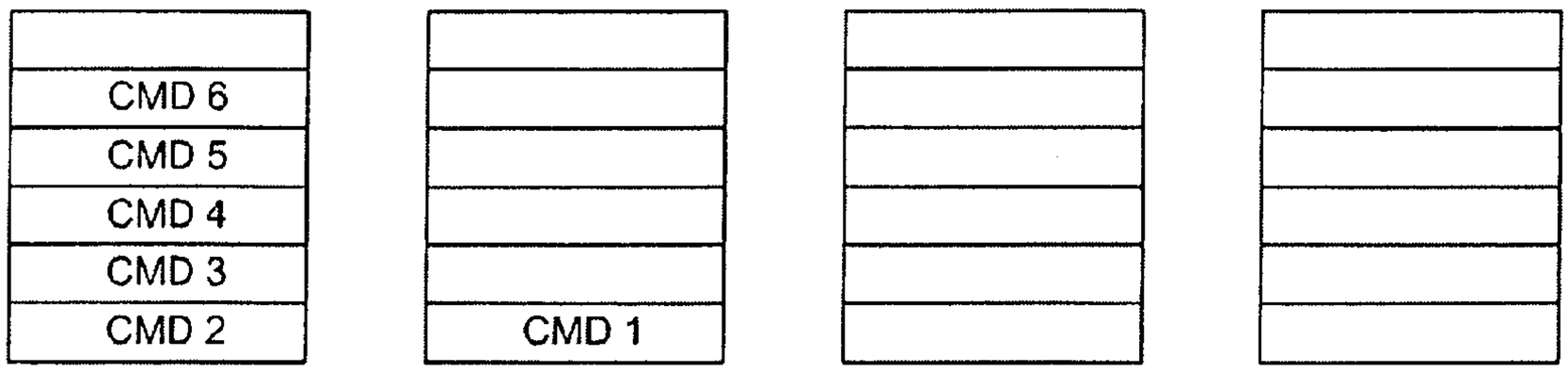


FIG. 3B

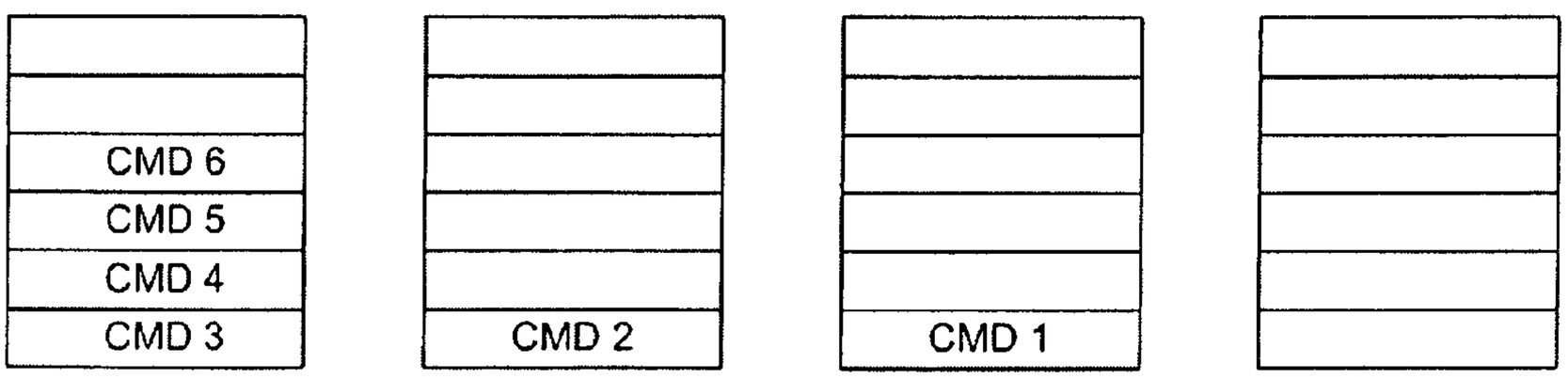


FIG. 3C

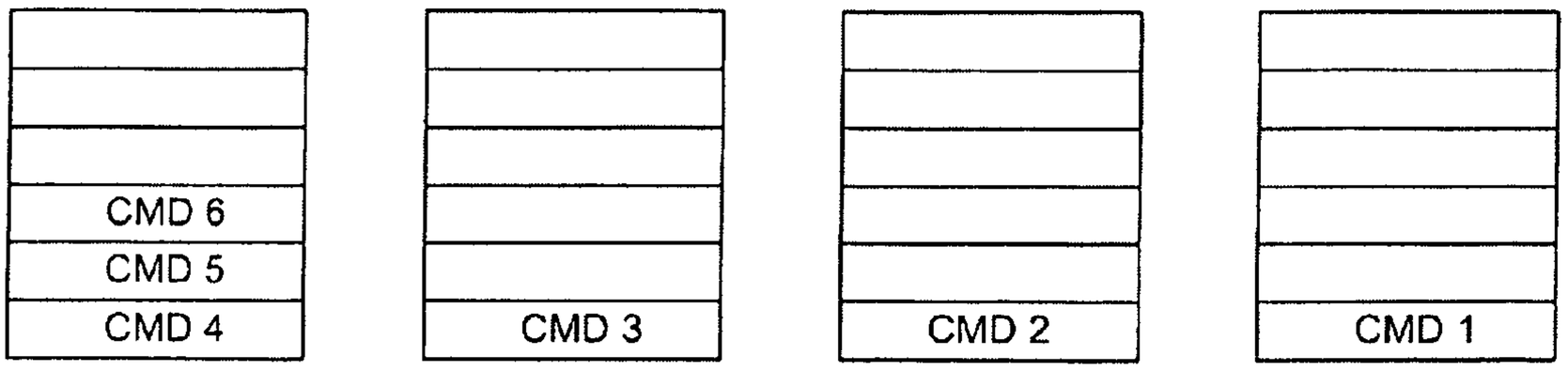


FIG. 3D

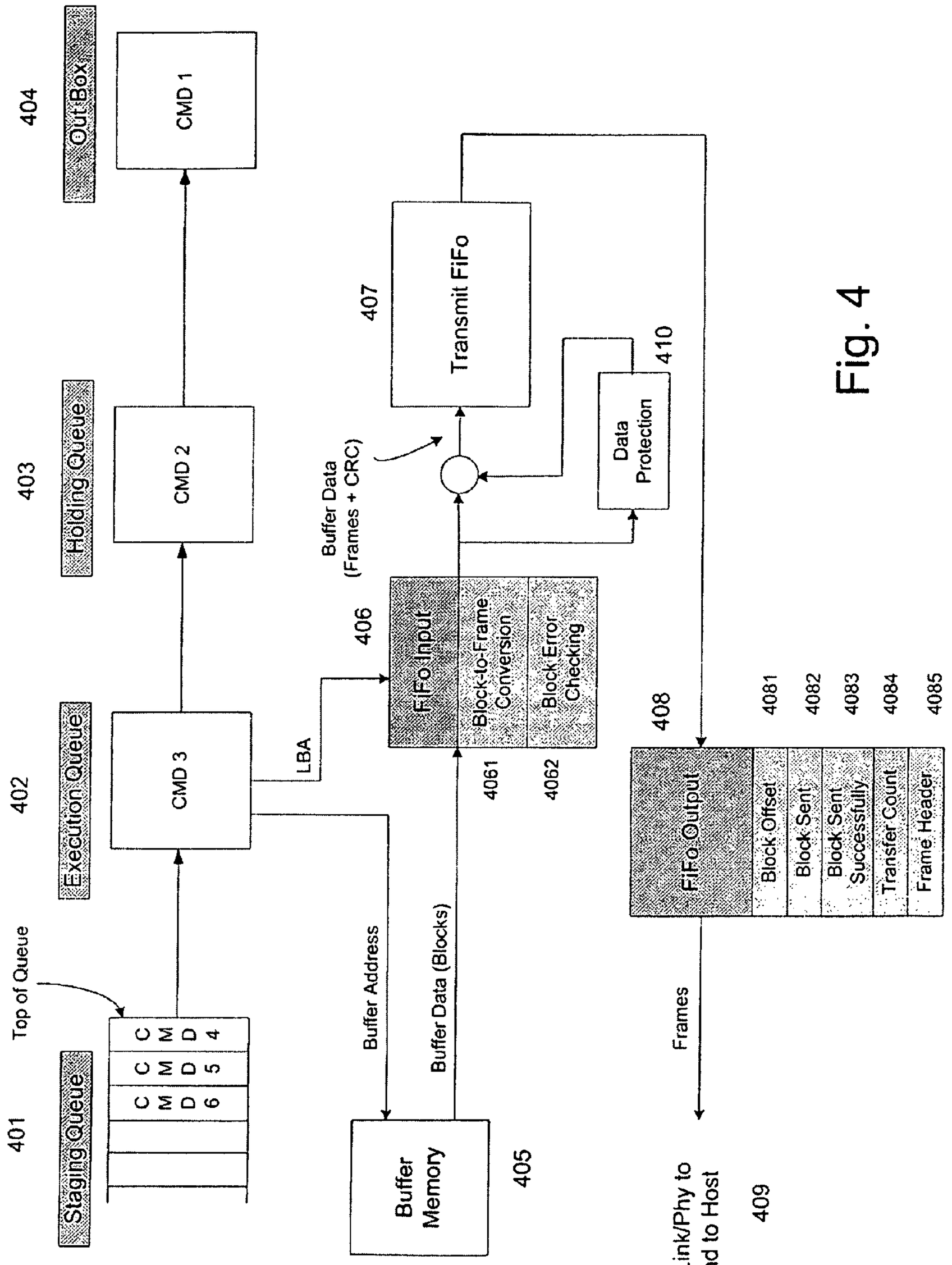

404
OutBox
CMD 1
403
Holding Queue
CMD 2
402
Execution Queue
CMD 3
LBA
406
FiFo Input
Block-to-Frame Conversion
Block Error Checking
4061
4062
Buffer Data (Frames + CRC)
407
Transmit FiFo
Data Protection
410
408
FiFo Output
Block Offset
Block Sent
Block Sent Successfully
Transfer Count
Frame Header
4081
4082
4083
4084
4085
Top of Queue
401
Staging Queue
C M D 6
C M D 5
C M D 4
Buffer Address
Buffer Data (Blocks)
Buffer Memory
405
Frames
To Link/Phy to send to Host
409

Fig. 4

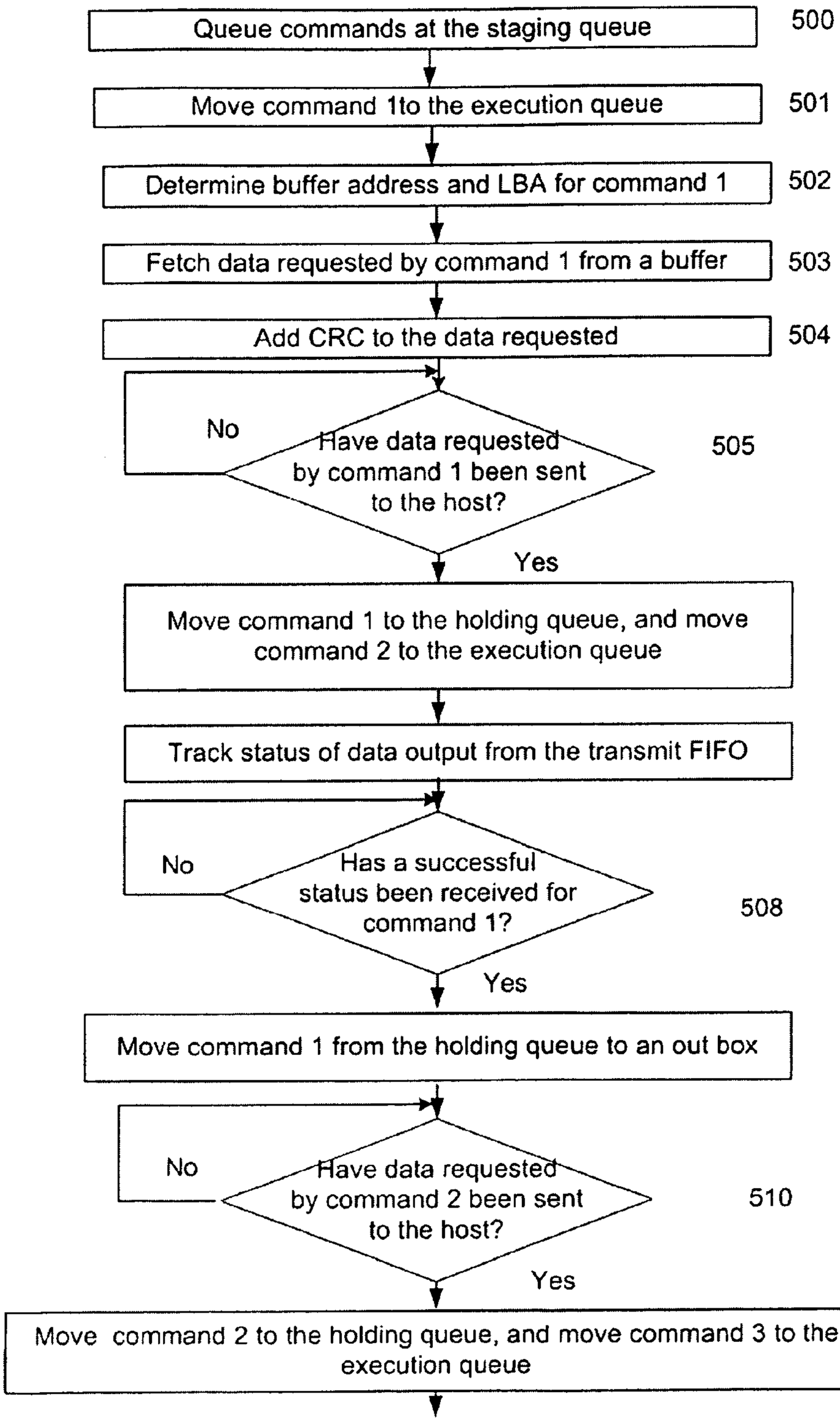

Queue commands at the staging queue
500
Move command 1to the execution queue
501
Determine buffer address and LBA for command 1
502
Fetch data requested by command 1 from a buffer
503
Add CRC to the data requested
504
No
Have data requested by command 1 been sent to the host?
505
Yes
Move command 1 to the holding queue, and move command 2 to the execution queue
506
Track status of data output from the transmit FIFO
507
No
Has a successful status been received for command 1?
508
Yes
Move command 1 from the holding queue to an out box
509
No
Have data requested by command 2 been sent to the host?
510
Yes
Move command 2 to the holding queue, and move command 3 to the execution queue
511

Fig. 5

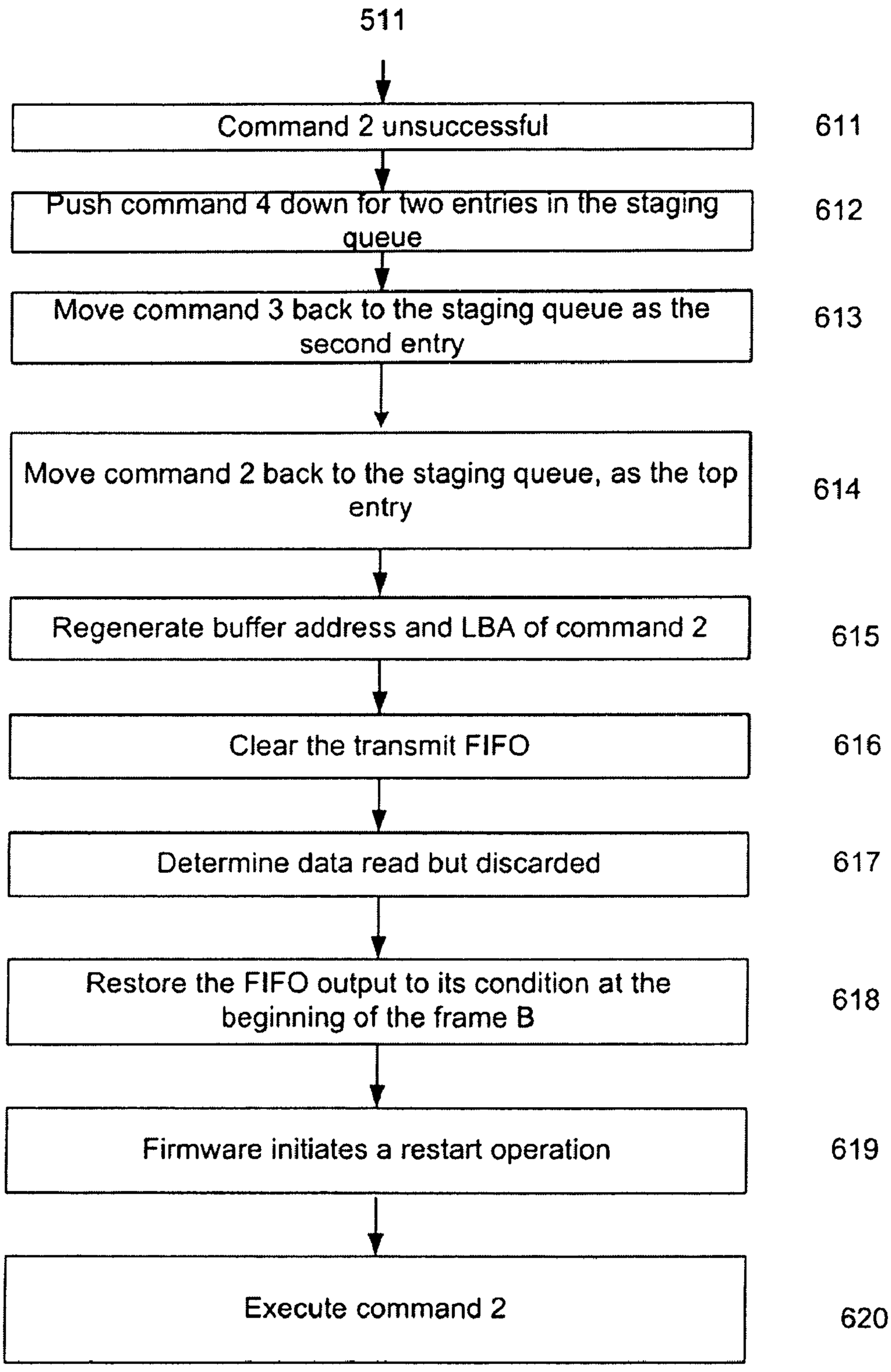

511
Command 2 unsuccessful
611
Push command 4 down for two entries in the staging queue
612
Move command 3 back to the staging queue as the second entry
613
Move command 2 back to the staging queue, as the top entry
614
Regenerate buffer address and LBA of command 2
615
Clear the transmit FIFO
616
Determine data read but discarded
617
Restore the FIFO output to its condition at the beginning of the frame B
618
Firmware initiates a restart operation
619
Execute command 2
620

Fig. 6

METHOD AND SYSTEM FOR COMMAND QUEUING IN DISK DRIVES

CROSS REFERENCE TO RELATED APPLICATION

This application claims the benefit of priority to previously filed U.S. provisional patent application Ser. No. 61/016,667, filed Dec. 26, 2007, entitled RESTART OPERATION IN QUEUED COMMANDS. That provisional application is hereby incorporated by reference in its entirety.

BACKGROUND

1. Field of the Invention

The present invention relates generally to disk drives, and more particularly to command queuing in disk drives.

2. Description of Related Art

In currently available disk drives, a controller (e.g., firmware) may issue a command for transferring some data to a host, hardware of the disk drive may execute the command, and the host may send a status report back to the firmware, indicating whether the command is executed successfully. The firmware may issue the command again if there is an error, or move to the next command if the transmission is successful. This process is not very efficient since the firmware needs to wait for the status report.

SUMMARY OF THE INVENTION

A method and system is disclosed for command queuing for disk drives which may improve performance by queuing multiple commands and automatically sequentially executing them without firmware intervention. The method may use a number of queues, e.g., a staging queue for commands to be executed, an execution queue for commands currently being executed, and a holding queue for commands which have been executed but have not received a status report from a host. With the pipelined nature of queued commands, when data requested by one command are being sent to the host, the queue logic may already be fetching data for the next command. If an error occurs during the transmission, commands in the queues may backtrack and restart from the point where data were last known to have been successfully sent to the host. Advantages of the present invention will become apparent from the following detailed description.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

Embodiments of the present invention are described herein with reference to the accompanying drawings, similar reference numbers being used to indicate functionally similar elements.

FIG. **1** illustrates a data stream to be transferred to a host.

FIG. **2** illustrates a flowchart of a command queuing operation according to one embodiment of the present invention.

FIGS. **3**A-**3**D illustrate queue status during a command queuing operation according to one embodiment of the present invention:

FIG. **4** illustrates a system for command queuing for a disk drive according to one embodiment of the present invention.

FIG. **5** illustrates a flowchart of a successful command queuing operation for a disk drive according to one embodiment of the present invention.

FIG. **6** illustrates a flowchart of a restart operation when there is an error in a command queuing operation according to one embodiment of the present invention.

DETAILED DESCRIPTION

FIG. **1** illustrates an exemplary data stream to be transferred to a host. As shown, each command may include a number of data blocks (e.g., blocks **1**-**8**), and each block may contain a number of bytes (e.g., 512 bytes). Data may be transmitted to a host as a series of frames (e.g., frames A, B and C). A frame size may have no correlation to a block size. For example, an FC (Fiber Channel) frame payload may have 0 to 2112 bytes, and an SAS (Serial-Attached Small Computer System Interface) frame payload may have 1 to 1024 bytes. In FIG. **1**, as an example, frame A may include blocks **1** and **2** and part of block **3**, and frame B may include part of block **3**, blocks **4** and **5**, and part of block **6**. The data stream shown in FIG. **1** may be used for FC or SAS.

FIG. **2** illustrates a flowchart of a command queuing operation according to one embodiment of the present invention, and FIGS. **3**A-**3**D illustrate queue status during a command queuing operation according to one embodiment of the present invention.

At **201**, firmware may write to a staging queue a number commands, e.g., commands **1**-**6**. The commands may request data transfer to a host. As shown in FIG. **3**A, command **1** is at the front of the staging queue.

At **202**, command **1** may initiate a request for data transfer and move to an execution queue, as shown in FIG. **3**B. Command **2** may move to the front of the staging queue.

At **203**, data requested by command **1** may be sent to a host.

At **204**, after all data requested by command **1** have been sent to the host, command **1** may move from the execution queue to a holding queue to wait for a status report from the host. As shown in FIG. **3**C, while command **1** is waiting for the status report in the holding queue, command **2** at the front of the staging queue may initiate a request for data transfer and move to the execution queue, and command **3** may percolate to the front of the staging queue. Command **1** may stay in the holding queue until the firmware receives a status report from the host. If all data for command **2** have been sent to the host before a status report for command **1** is received by the firmware, transmission may stop with command **2** in the execution queue and command **3** at the front of the staging queue.

If the firmware receives a successful status report for command **1** from the host at **205**, command **1** may move to an out box at **206**. Meanwhile, if the data requested by command **2** have been transferred to the host at **207**, at **208**, command **2** may move to the holding queue to wait for a status report there, command **3** may initiate a request for data transfer and move to the execution queue, and command **4** may percolate to the front of the staging queue, as shown in FIG. **3**D. Thus, when command **1** is waiting for its status report, the hardware of the disk drive may execute command **2**, thus reducing the waiting time and improving performance of the disk drive.

In one embodiment, command **2** may move to the holding queue before command **1** leaves the holding queue. In one embodiment, more commands may be put into the execution queue and/or the holding queue, as long as the command in the holding queue whose status is unsuccessful can be put back to the staging queue as the front entry during a restart operation. In one embodiment, **205** and **207** may happen simultaneously, and **206** and **208** may happen simultaneously.

FIG. **4** illustrates a system for command queuing for a disk drive according to one embodiment of the present invention. The system may have a first memory **401** for a staging queue, a second memory **402** for an execution queue, and a third memory **403** for a holding queue. In one embodiment, the memories **401-403** may be FIFOs (First-In-First-Out). In one embodiment, the memories **401-403** may be part of the same memory device. When a command is completed without error, it may move to an out box **404**. The data to be transmitted may be read from a buffer memory **405** in a disk drive. When a command is being executed; the buffer address of data blocks to be transmitted may be used to locate the data in the buffer memory **405**, and the data may be transmitted to a host via an FIFO input **406** of a transmit FIFO **407**, the transmit FIFO **407**, an FIFO output **408** of the transmit FIFO **407**, and a Link/Phy layer **409**.

The FIFO input **406** may receive data from the buffer memory **405**. In one embodiment, the buffer data may be received in blocks. The FIFO input **406** may have a block-to-frame conversion module **4061** for converting the buffer data from blocks into frames, and a block error checking module **4062** for checking if there is any error in a data block.

A data protection module **410** may be used for data integrity check. In one embodiment, a CRC (Cyclic Redundancy Check) word may be added to the data frames from the FIFO input **406**.

The FIFO output **408** may track information of successfully transmitted data, so that if there is an error in the data transmission, the system may accurately backtrack and restart from the point where data were last known to have been successfully sent to the host. The FIFO output **408** may keep the following values during the operation: a block offset **4081**, a number of blocks sent **4082**, a number of blocks sent successfully **4083**, a number of bytes to transfer **4084**, and a frame header **4085**.

The number of blocks sent **4082** may track the number of blocks sent but not acknowledged as received error-free at the host. The host may not acknowledge each frame as it arrives but may accumulate many frames before sending the acknowledgement. These frames may have their block count accounted for in **4082**, and when the acknowledgement eventually arrives, the value in **4082** may be used to update the number of blocks sent successfully **4083**. If an error occurs in the transmission since the last acknowledgement, the value in **4082** may be simply discarded.

The number of bytes to transfer **4084** may track the amount of data sent to the host. It may double-check the amount of data gathered from the buffer **405** through the FIFO input **406** and the Transmit FIFO **407**.

The Relative Offset/parameter field in the Frame Header **4085** may identify where in the whole transfer the data in this frame belongs and may be updated as each byte is sent to the host.

Each new command may initiate the following operations in the FIFO output box **408**:

a) The block offset **4081** may be set to the block size of data in the command;

b) The number of blocks sent **4082** may be cleared to zero;

c) The number of blocks sent successfully **4083** may be cleared to zero;

d) The number of bytes to transfer **4084** may be set to the transfer size in bytes; and

e) A Relative Offset/Parameter field in the frame header **4085** may be set to zero or to an initial value by the firmware for the first command, or may be a continuation of the value from the previous command

As each word of payload leaves the transmit FIFO **407**, the following changes may take place in the FIFO output **408**:

a) The block offset **4081** may decrement by the amount of data transmitted in a block. Once it reaches zero, it is reloaded with the size of the data block in the command;

b) The number of blocks sent **4082** may increment by 1 each time the block offset **4081** counts down to zero;

c) The number of blocks sent successfully **4083** may be updated by the number of blocks sent in a frame which the host has indicated being received error-free;

d) The number of bytes to transfer **4084** may decrement by the amount of data transmitted. Once it reaches zero, all data for the command have been sent; and

e) The Relative Offset/Parameter field may increment by the amount of data.

FIG. **5** illustrates a flowchart of a successful command queuing operation for a disk drive according to one embodiment of the present invention. The method may be used in the system shown in FIG. **4**.

Firmware may write commands **1-6** to a staging queue at **500**. Each command may include: an initial buffer address, an initial LBA (Logical Block Address), a Skip LBA (number of LBA to skip) and an integral number of blocks to transfer. The block size, e.g., in bytes, may be a static value for the whole operation, and may not be a part of the command. The block size and number of blocks may be used to generate the number of bytes to transfer **4084**.

At **501**, command **1** may bubble up to the front of the staging queue, initiate a request for data transfer and move to the execution queue.

At **502**, a buffer address and an LBA may be generated for command **1** based on the following equations:

$$\text{Buffer address}=\text{initial buffer address}+(\text{block size})*(\text{Skip LBA}) \quad (1)$$

$$\text{LBA}=\text{initial LBA}+\text{Skip LBA} \quad (2)$$

Data to be transferred may be read from a location in the buffer memory **405** pointed to by the buffer address and the LBA may be used as the seed to check integrity of data coming from the buffer memory **405**. The initial buffer address, initial LBA, Skip LBA and number of blocks to transfer may be saved in the execution queue.

At **503**, data blocks requested by command **1** may be fetched from the buffer memory **405** and sent to the transmit FIFO **407**. The FIFO input **406** may convert incoming data, in blocks, into data in the size of a designated frame payload.

At **504**, a CRC word may be added by the data protection module **410** to each frame payload from the FIFO input **406** to aid in error detection.

If all data requested by command **1** have been sent to the transmit FIFO **407** at **505**, then command **1** may move from the execution queue in the memory **402** to the holding queue in the memory **403** at **506**. Command **1**'s initial buffer address, initial LBA, Skip LBA and number of blocks to transfer may also move from the execution queue to the holding queue.

At the same time, command **2** may move from the staging queue to the execution queue, and data requested by command **2** may start to be fetched.

At **507**, at the FIFO output **408**, each frame payload of data leaving the FIFO **407** may be preceded by a header and sent to the Link/Phy **409** on its way to the host. The FIFO output **408** may track the block offset **4081** in a block; track the number of blocks sent **4082**; track the number of blocks sent successfully **4083**; update the number of bytes to transfer **4084**; and update the frame header **4085** for the next frame.

If the host acknowledges that all data for command **1** have been successfully received at **508**, command **1** may move from the holding queue to an out box at **509**, where it may be serviced/discarded by the hardware or examined by the firmware. Meanwhile, if data transfer for command **2** is completed at **510**, at **511**, command **2** may go to the holding queue, command **3** may go to the execution queue, and command **4** may percolate to the front of the staging queue. FIGS. **3**D and **4** show the status of the commands at this moment.

A host may not always receive the data correctly, e.g., when a frame is lost or corrupted in transmission. In this scenario, the command queuing operation may have to be suspended and a restart operation may need to begin to transfer the data which were not successfully transmitted. FIG. **6** illustrates a flowchart of a restart operation when there is an error in a command queuing operation according to one embodiment of the present invention.

The process may follow **511**. At **511**, command **1** may receive a successful status and move to the out box **404**, all data for command **2** have been sent to the transmit FIFO **407** and command **2** may move to the holding queue, command **3** may initiate a request for data transfer and move to the execution queue, and command **4** may percolate to the front of the staging queue.

At **611**, while data requested by command **3** are being sent to the transmit FIFO **407**, the firmware may receive from the host an unsuccessful status for command **2**. For example, frame B shown in FIG. **1** may have a transmission error. Since frame A has been transmitted successfully, blocks **1** and **2** have been received by the host, and the beginning part of block **3** may have been received successfully too. But the remaining part of block **3**, blocks **4** and **5**, and the beginning part of block **6** may have transmission error.

At **612**, data transmission for command **3** may stop and the operation may return to the beginning of command **2**. The content of the staging queue may be pushed down by two entries, i.e. command **4** may be pushed from the front of queue to the third entry from the front of queue.

At **613**, command **3** in the execution queue may be written back to the staging queue, as the second entry from the front of the queue.

At **614**, command **2** in the holding queue may be put back into the staging queue as the front entry, together with its initial buffer address, initial LBA, Skip LBA and number of blocks to transfer.

At **615**, the buffer address and LBA for command **2** may be regenerated according to equations (1) and (2). In one embodiment, since the host has indicated that frame A was received successfully, the buffer address and LBA may be adjusted for blocks **1** and **2** that were sent successfully, so that they will not be sent again. The adjusted LBA may be used to seed the data integrity check logic.

At **616**, the pipeline and transmit FIFO **407** may be cleared of data.

At **617**, the block offset **4081** from the FIFO output **408** may determine the amount of data in block **3** that were read but discarded. In one embodiment, data may be fetched from the buffer memory **405** from the beginning of block **3** to satisfy the data integrity check requirements, but only data after the block offset **4081** may be resent to the transmit FIFO **407**.

At **618**, the FIFO output **408** may be restored to its condition at the beginning of frame B. The values of the block offset **4081**, the number of blocks sent **4082**, the number of blocks sent successfully **4083**, the number of bytes to transfer **4084**, and the frame header **4085** may be restored exactly as when frame B was last built. The number of bytes to transfer **4084** may be generated based on the following formular:

((number of blocks to transfer−number of blocks sent successfully 4083)×block size)−block offset 4081

The Relative Offset/Parameter value may be generated based on the following formula:

Initial Relative Offset/Parameter+(number of blocks sent successfully 4083×block size)+block offset 4081

At **619**, the firmware may initiate a restart operation so that the hardware knows to check for the block offset **4081**, and the number of blocks sent successfully **4083**, as opposed to a start operation where such values do not need to be checked.

At **620**, command **2** may move from the front of the staging queue to the execution queue and the data transmission process may restart from the beginning of frame B.

In addition to disk drives, the present invention may also be used in other storage devices, e.g., solid state drives. Accordingly, as used herein the term "disk drive" includes solid state drives.

Several features and aspects of the present invention have been illustrated and described in detail with reference to particular embodiments by way of example only, and not by way of limitation. Alternative implementations and various modifications to the disclosed embodiments are within the scope and contemplation of the present disclosure. Therefore, it is intended that the invention be considered as limited only by the scope of the appended claims.

What is claimed is:

1. A system for command queuing for a disk drive, comprising:

a first queue for storing commands to be executed, wherein the commands are for transferring data between the disk drive and a host;

a second queue for storing a first command received from the first queue when the first command is being executed; and

a third queue for storing the first command received from the second queue after the first command has been executed but before a status report for the first command is available.

2. The system of claim **1**, further comprising a memory device for temporarily storing data transmitted between the disk drive and the host pursuant to the commands.

3. The system of claim **2**, wherein the memory device has an input control for converting data from blocks into frames.

4. The system of claim **2**, further comprising a data protector for adding error check information to an input of the memory device.

5. The system of claim **4**, wherein the error check information is a CRC (Cyclic Redundancy Check) word.

6. The system of claim **2**, wherein the memory device has an output control for tracking data output of the memory device.

7. The system of claim **6**, wherein the output control tracks a block offset in a data block from the memory device.

8. The system of claim **6**, wherein the output control tracks a number of blocks sent from the memory device.

9. The system of claim **6**, wherein the output control tracks a number of blocks sent successfully from the memory device.

10. The system of claim **6**, wherein the output control tracks a number of bytes to transfer in a command.

11. The system of claim **6**, wherein the output control keeps a frame header.

12. A method for command queuing for a disk drive, comprising:

queuing at least a first command and a second command in a first queue, wherein the first and second commands are for transferring data between the disk drive and a host;

moving the first command from the first queue to a second queue for execution;

moving the first command from the second queue to a third queue after the first command has been executed but before the host returns a status report; and

moving a second command from the first queue to the second queue for execution when the first command is in the third queue.

13. The method of claim 12, further comprising: temporarily storing data transferred between the host and the disk drive in a memory device pursuant to the first and second commands.

14. The method of claim 13, further comprising: converting data from blocks into frames.

15. The method of claim 13, further comprising: removing the first command from the third queue when the host has confirmed that a data transfer requested by the first command is successful, and moving the second command from the second queue to the third queue.

16. The method of claim 13, further comprising: moving the first command back to the first queue when the host indicates that there is an error in the data transfer requested by the first command.

17. The method of claim 13, further comprising: repeating execution of the first command.

18. The method of claim 17, further comprising: determining part of data requested by the first command that have been successfully transferred.

19. The method of claim 18, further comprising: resending part of data requested by the first command that have not been successfully transferred.

20. The method of claim 13, further comprising: tracking a block offset in a data block from the memory device.

21. The method of claim 13, further comprising: tracking a number of blocks sent successfully from the memory device.

22. The method of claim 13, further comprising:

queuing a third command in the first queue;

moving the third command from the first queue to the second queue for execution;

moving the third command from the second queue to the third queue after the third command has been executed but before the second command leaves the third queue.

23. The method of claim 13, further comprising: adding error check information to an input of the memory device.

24. The method of claim 23, wherein the error check information is a CRC (Cyclic Redundancy Check) word.

* * * * *